Projeto e Análise de Algoritmos

Aula 03 – Técnicas de Projeto de Algoritmos (Divisão e Conquista)

Edirlei Soares de Lima

<edirlei@iprj.uerj.br>

Estratégias de Projeto de Algoritmos

- **Força Bruta (Brute Force)**
- **Dividir e Conquistar (Divide and Conquer)**
- Diminuir e Conquistar (Decrease and Conquer)
- Transformar e Conquistar (Transform and Conquer)
- Compromisso Tempo–Espaço (Space and Time Tradeoffs)
- **Estratégia Gulosa (Greedy)**
- **Programação Dinâmica (Dynamic Programming)**
- Voltando Atrás (Backtracking)
- Ramificar e Limitar (Branch and Bound)
- Algoritmos Aproximados

Divisão e Conquista

- **Ideia geral:**

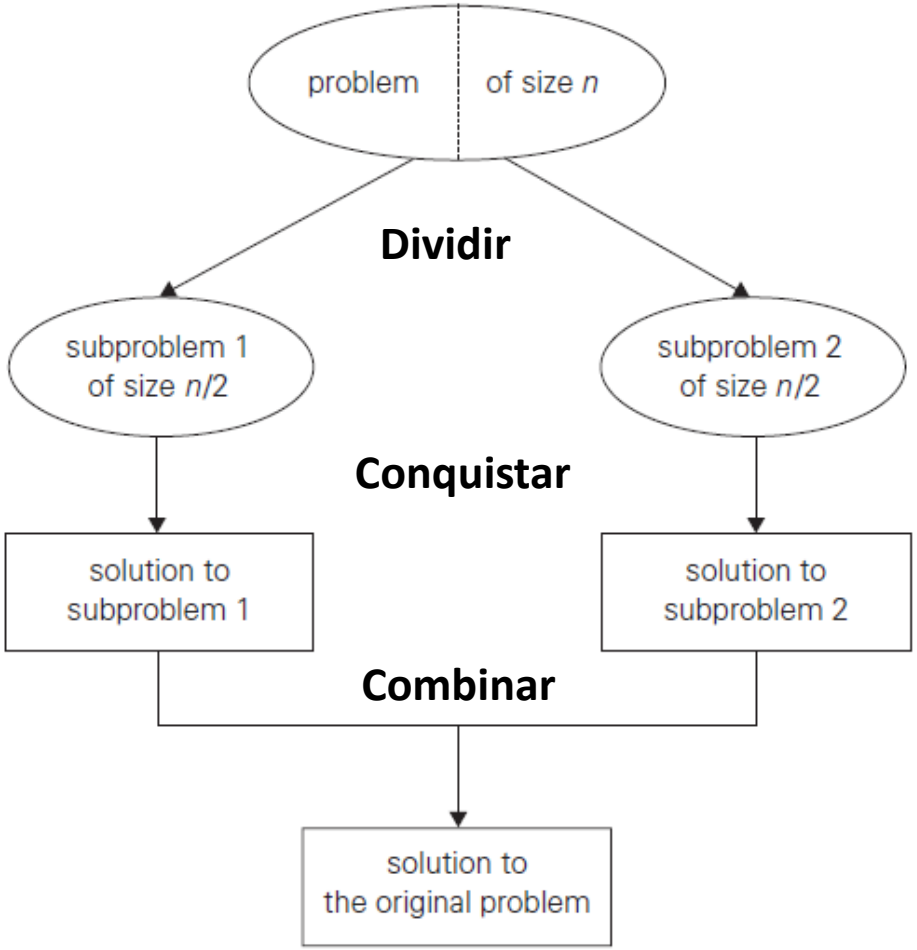
1. Dividir a instância do problema em duas ou mais instâncias menores;

2. Resolver as instâncias menores (geralmente recursivamente);

3. Obter a solução para as instâncias originais (maiores) através da combinação destas soluções.

Divisão e Conquista

- O paradigma de dividir e conquistar envolve 3 passos:



Divisão e Conquista

- **Exemplo:** calcular a soma de n números $a_0 + \ \ldots + a_{n-1}$

- Se n > 1, podemos dividir o problema em duas instâncias do mesmo problema:
  - Soma dos primeiros $\lfloor n/2 \rfloor$ números;
  - Soma dos $\lceil n/2 \rceil$ números restantes;

- Uma vez estas duas somas computadas, adicionamos seus valores para obter o resultado final:

$$a_0 + \ \ldots + a_{n-1} = (a_0 + \ \ldots + a_{\lfloor n/2 \rfloor - 1}) + (a_0 + \ \ldots + a_{\lceil n/2 \rceil - 1})$$

Divisão e Conquista

- Esta é uma maneira eficiente de calcular a soma de n números?
- É mais eficiente do que uma adição força bruta?
- Nem todos os algoritmos baseados na técnica de dividir e conquistar são mais eficientes do que uma solução força bruta.

Divisão e Conquista

- Em geral, o tempo gasto na execução das três etapas do algoritmo dividir e conquistar, é menor do que a resolução por outros métodos.

- A estratégia dividir e conquistar produz alguns dos **algoritmos mais importantes e eficientes** da área da computação.

- Importante: a estratégia dividir e conquistar pode ser facilmente adaptada a **computação paralela**.

Exemplo: Multiplicação de Inteiros Grandes

- **Problema:** realizar a multiplicação de inteiros grandes (da ordem de 100 dígitos decimais).
  - Ex: 35871225379742800 9 × 15346123490876438 8 = ?

Multiplicação Tradicional:

$$
\begin{array}{rl}
9999 & u \\
7777 & v \\
\hline
69993 & \\
69993 & \\
69993 & \\
69993 & \\
\hline
77762223 & u \times v
\end{array}
$$

Algoritmo de Força Bruta:

- multiplicação dígito a dígito;
- soma dígito a dígito;

Complexidade: $O(n^2 + n) = \mathbf{O(n^2)}$

Exemplo: Multiplicação de Inteiros Grandes

- **Problema:** realizar a multiplicação de inteiros grandes (da ordem de 100 dígitos decimais).
  - Outra forma de multiplicar números grandes?

Passo 1 - Multiplicação dos Grandes

$A = x_1 \times y_1$ $A = 12 \times 56 = 672$

Passo 2 - Multiplicação dos Pequenos

$B = x_2 \times y_2$ $B = 34 \times 78 = 2652$

Passo 3 - Soma os dois grupos de x

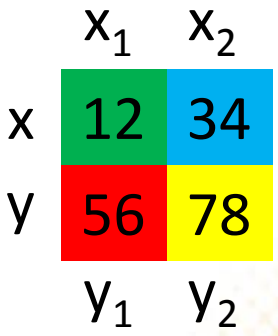
$C = x_1 + x_2$ $C = 12 + 34 = 46$

Passo 4 - Soma os dois grupos de y

$D = y_1 + y_2$ $D = 56 + 78 = 134$

Passo 5 - Multiplique as somas dos grupos

$E = C \times D$ $E = 46 \times 134 = 6164$



Exemplo: Multiplicação de Inteiros Grandes

- **Problema:** realizar a multiplicação de inteiros grandes (da ordem de 100 dígitos decimais).
  - Outra forma de multiplicar números grandes?

Passo 6 - Subtraia o produto da soma dos dois grupos (E), o produto dos grandes (A) e o produto dos pequenos (B)

$F = E - A - B$ $\quad F = 6164 - 672 - 2652 = 2840$

Passo 7 - Shift do produto dos grandes

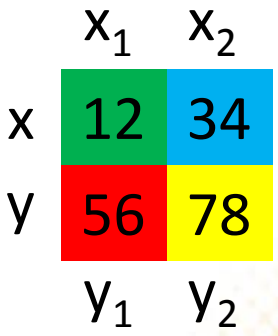
$G = A \times 10^{2m}$ $\quad G = 672 \times 10^{2x2} = 6720000$

Passo 8 - Shift do resultado das subtrações

$H = F \times 10^{m}$ $\quad H = 2840 \times 10^{2} = 284000$

Passo 9 - Soma Final

$I = G + B + H$ $\quad I = 6720000 + 2652 + 284000$



Exemplo: Multiplicação de Inteiros Grandes

- Algoritmo de Karatsuba

```
1. KARATSUBA(u, v, n)
2.   se n < 3 então
3.     retorne u × v
4.   senão
5.     m ← ⌈n/2⌉
6.     a ← ⌊u/10^m⌋
7.     b ← u mod 10^m
8.     c ← ⌊v/10^m⌋
9.     d ← v mod 10^m
10.    ac ← KARATSUBA(a, c, m)
11.    bd ← KARATSUBA(b, d, m)
12.    y ← KARATSUBA(a + b, c + d, m + 1)
13.    x ← ac × 10^2m + (y - ac - bd) × 10^m + bd
14.    retorne x
```

$O(n^{log_2 3}) = O(n^{1.585})$

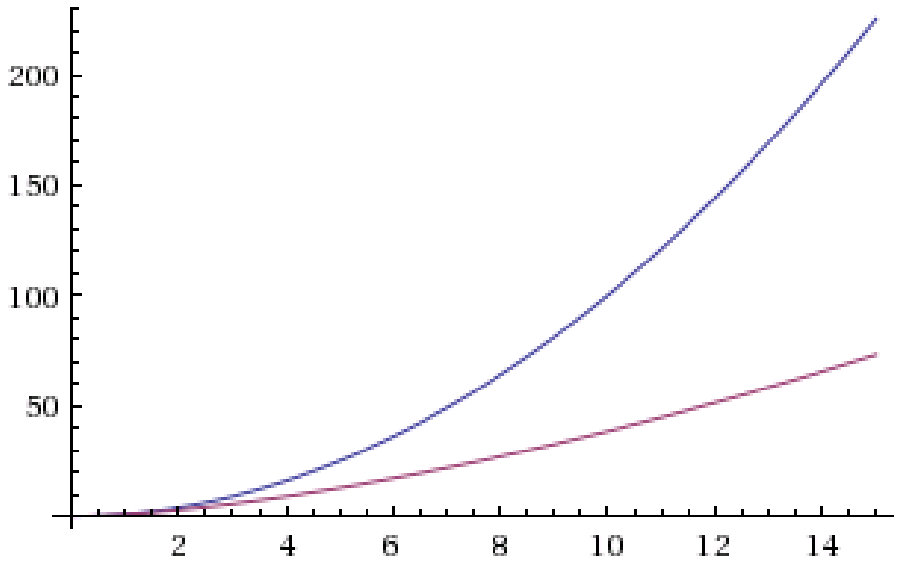
Exemplo: Multiplicação de Inteiros Grandes

- Multiplicação Tradicional: O(n^2)
- Algoritmo de Karatsuba: O($n^{log_2 3}$)

Wolfram Alpha:

```
plot n^2 from n=0 to 15, n ^ log base 2 of 3 from n=0 to 15
```

Plot:



Divisão e Conquista

- Outros algoritmos importantes baseados em divisão e conquista que veremos ao longo do curso:
  - Merge Sort
  - Quick Sort

Quando Utilizar Divisão e Conquista?

- Existem três condições que indicam que a estratégia de divisão e conquista pode ser utilizada com sucesso:
  - Deve ser possível **decompor** uma instância em sub-instâncias.
  - A **combinação** dos resultados dever ser eficiente (trivial se possível).
  - As sub-instâncias devem ser mais ou menos do **mesmo tamanho**.

Comentários sobre a Divisão e Conquista

- Muitos algoritmos eficientes são baseados nesta técnica.
  - Contudo, ela pode ser inaplicável e inferior a soluções algorítmicas mais simples.

- Outras estratégias semelhantes:
  - Diminuir e Conquistar (Decrease and Conquer)
  - Transformar e Conquistar (Transform and Conquer)

Comentários sobre a Divisão e Conquista

- Vantagens:
  - Pode gerar algoritmos eficientes com forte tendência a complexidade logarítmica;
  - Facilmente paralelizável;

- Desvantagens:
  - Geralmente são algoritmos recursivos (problema de estouro de pilha);
  - Repetição de sub-problemas;

Exercícios

Lista de Exercícios 03 – Divisão e Conquista

http://www.inf.puc-rio.br/~elima/paa/

Leitura Complementar

- Levitin. **Introduction to the Design and Analysis of Algorithms**, 3rd Edition, 2011.

- **Capítulo 5: Divide-and-Conquer**